Linux期末考试

23物联网工程 2023124306

1.P4：Linux内核版本号r.x.y的含义。（单选）

1. - **r**：表示主版本号（Major version）。这个数字代表内核的重大版本更新。当内核进行大规模的功能变化、架构重构或者不兼容的更改时，主版本号会增加。
   - **x**：表示次版本号（Minor version）。次版本号的变化通常表示内核进行了一些较小的改进或增加了新功能，但通常不会改变系统的兼容性。
   - **y**：表示修订号（Patch level）。修订号用于表示内核的小的修复或安全补丁更新，这些更改不会影响内核的功能或兼容性。

2.P15：快捷键。快捷键Ctrl+D、Ctrl+Z和Ctrl+C的作用。（单选）

1. **Ctrl+D**：
   - **作用**：发送 **EOF**（文件结束）信号，通常用于结束输入或退出终端会话。
   - **场景**：例如在命令行中使用 `Ctrl+D` 可以退出 `cat`、`python` 等交互式命令行模式，或者结束一个终端会话。
2. **Ctrl+Z**：
   - **作用**：将当前正在运行的进程 **挂起**（暂停），并将其置于后台。
   - **场景**：例如，在执行一个命令时按下 `Ctrl+Z` 会暂停该命令的执行，可以通过 `fg` 恢复到前台继续执行。
3. **Ctrl+C**：
   - **作用**：发送 **SIGINT**（中断）信号，终止当前运行的命令或进程。
   - **场景**：例如，在执行某个命令时按下 `Ctrl+C` 可以立即中止该命令的执行

3.P17：help,whatis, whereis,man命令的用途。（单选）

- **help**：查看内建命令的帮助信息。
- **whatis**：查看命令的简短描述。
- **whereis**：查找命令的二进制文件、源代码和手册页的位置。
- **man**：查看命令的详细手册。

4.P31：*、?、[]、[!]等文件名通配符的用法，参考表2-5。（单选）

- ***：匹配零个或多个字符。

- **?**：匹配一个任意字符。
- **[]**：匹配方括号内的任意一个字符。
- **[!]**：匹配不在方括号中的字符。

5.P31：ls -l命令显示的文件类型（输出的每行的第一个字符），能识别出-、d、l、c、b、s、p分别表示什么类型的文件。单选）

- **-**：普通文件
- **d**：目录
- **l**：符号链接
- **c**：字符设备文件
- **b**：块设备文件
- **s**：套接字文件
- **p**：命名管道

6.P35：Linux文件系统布局。根目录下boot, dev, etc, home, proc,

- **/boot**：系统引导相关的文件。
- **/dev**：设备文件。
- **/etc**：系统配置文件。
- **/home**：用户数据文件。
- **/proc**：系统信息文件。**Process** (进程)
- **/var**：可变数据文件（日志、缓存等）。**Variable** (可变)

7.P36：命令ls,touch, cp, mv, rm, ln,pwd, cd, find, mkdir, rmdir命令的用途。（单选）

- **ls**：列出目录内容。
- **touch**：创建空文件或更新文件时间戳。
- **cp**：复制文件或目录。
- **mv**：移动或重命名文件/目录。
- **rm**：删除文件或目录。
- **ln**：创建文件链接（符号链接或硬链接）。
    - ▪ `-s`：创建符号链接（软链接），而不是硬链接。符号链接类似于快捷方式。
    - **例子**：`ln -s source target` 创建一个指向 `source` 的符号链接 `target`。
- **pwd**：显示当前工作目录路径。
- **cd**：切换当前工作目录。
- **find**：在目录树中查找文件。
- **mkdir**：创建新目录。
- **rmdir**：删除空目录。

8.P39：命令cat, less, more, head, tail, cut, wc, sort,uniq,tr的用途。（单选）

- **cat**：连接文件并显示内容。
- **less**：分页显示文件内容，支持滚动。
- **more**：分页显示文件内容，支持向下滚动。
- **head**：显示文件的前几行。
- **tail**：显示文件的最后几行。
- **cut**：按列剪切文本内容。
  - **用途**：按列剪切文本内容，常用于从以分隔符分割的文件中提取特定的字段。
  - **例子**：`cut -d',' -f1 file.csv` 提取 `file.csv` 中以逗号分隔的第一列。
- **wc**：统计行数、字数和字符数。
- **sort**：对文件内容进行排序。
  - **用途**：对文件内容进行排序。默认按字典顺序排序。
  - **例子**：`sort file.txt` 对 `file.txt` 的内容进行排序
- **uniq**：去除重复行。
  - **用途**：去除文件中的重复行，通常和 `sort` 配合使用。
  - **例子**：`sort file.txt | uniq` 排序并去除 `file.txt` 中的重复行
- **tr**：转换字符或删除字符。 **translate**
  - **用途**：转换字符或删除字符，可以用于字符替换、删除等操作。
  - **例子**：`echo "hello" | tr 'a-z' 'A-Z'` 将 "hello" 转换为大写 "HELLO"

9.正则表达式，egrep命令

看egrep命令写执行结果（写命令执行结果，共4小题12分），只用写输出多少行，例如写3，表示输出3行，不用写具体输出的是哪几行。

常见符号及含义：

1. **.（点）**：匹配任意单个字符（除了换行符）。
   - 例：`a.b` 会匹配 `a` 后面跟一个字符，再跟一个 `b`，如 `axb`、`acb` 等。
2. **[ ]（方括号）**：匹配方括号内的任意一个字符。
   - 例：`[aeiou]` 会匹配任何一个元音字母。
3. **[^]（方括号内的插入^）**：匹配不在方括号中的任意字符。
   - 例：`[^aeiou]` 会匹配任何非元音字母的字符。
4. **?**：匹配前面的字符0次或1次。
   - 例：`colou?r` 会匹配 `color` 或 `colour`。
5. ***：匹配前面的字符0次或多次。
   - 例：`ab*c` 会匹配 `ac`、`abc`、`abbc` 等。
6. **+**：匹配前面的字符1次或多次。

- 例：`ab+c` 会匹配 `abc`、`abbc` 等，但不会匹配 `ac`。

7. **{n}**：匹配前面的字符恰好n次。
   - 例：`a{3}` 会匹配 `aaa`。
8. **{n,}**：匹配前面的字符至少n次。
   - 例：`a{2,}` 会匹配 `aa`、`aaa`、`aaaa` 等。
9. **{,m}**：匹配前面的字符最多m次。
   - 例：`a{,3}` 会匹配 `a`、`aa`、`aaa`。
10. **{n,m}**：匹配前面的字符至少n次，至多m次。
    - 例：`a{2,4}` 会匹配 `aa`、`aaa`、`aaaa`。
11. **：转义字符，将特殊符号变为普通字符。
    - 例：`\.` 会匹配 `.` 字符本身，而不是任何字符。
12. **( )**：圆括号用于分组，表示一组字符。
    - 例：`(ab)+` 会匹配 `ab`、`abab`、`ababab` 等。
13. **^**：匹配行的开头。
    - 例：`^abc` 会匹配以 `abc` 开头的行。
14. **$**：匹配行的结尾。
    - 例：`abc$` 会匹配以 `abc` 结尾的行。
15. **<**：匹配单词的开头。
    - 例：`\bcat` 会匹配单词 `cat` 的开头。
16. **>**：匹配单词的结尾。
    - 例：`cat\b` 会匹配单词 `cat` 的结尾。
17. **|**：表示"或"操作符，匹配左右任意一部分。
    - 例：`abc|def` 会匹配 `abc` 或 `def`。

10.find命令的用法

解释命令的含义/用途。（简答）

常见参数：

- **-maxdepth**：指定最大查找深度。例如，`-maxdepth 3` 表示只查找当前目录及其下的子目录的前三层。
- **-name**：按文件名进行匹配。
- **-perm**：按文件权限查找。
- **-uid**：按文件的用户 ID 查找。
- **-mtime**：按文件的修改时间查找。
- **-atime**：按文件的访问时间查找。
- **-ctime**：按文件属性修改时间查找。
- **-size**：按文件大小查找

示例解释：

1. **find . -name "*.txt" -print**
   - **含义**：在当前目录（`.`）下查找所有文件名以 `.txt` 结尾的文件并显示它们的路径。
   - **用途**：查找文本文件。
2. **find . -mtime -2 -print**
   - **含义**：在当前目录下查找最近 2 天内修改过的文件并显示。
   - **用途**：查找最近修改的文件，`-mtime -2` 表示过去 2 天内修改过的文件。
3. **find /home -perm 777 -print**
   - **含义**：在 `/home` 目录下查找权限为 777（即所有者、组和其他人都有读、写、执行权限）的文件并显示。
   - **用途**：查找具有特定权限的文件。
4. **find /home -atime +100 -print**
   - **含义**：在 `/home` 目录下查找访问时间超过 100 天的文件并显示。
   - **用途**：查找很久没有被访问的文件。
5. **find /home -size +50M**
   - **含义**：在 `/home` 目录下查找文件大小大于 50MB 的文件并显示。
   - **用途**：查找大文件，例如清理占用磁盘空间的文件。
6. **find /var -name "*.tmp" -exec rm {} \;**
   - **含义**：在 `/var` 目录下查找所有文件名以 `.tmp` 结尾的文件，并删除它们。`{}` 是 `find` 命令找到的每个文件的占位符，`\;` 用于结束 `-exec` 命令。
   - **用途**：删除临时文件。
7. **find ~ -maxdepth 3 -ctime -30**
   - **含义**：在用户的家目录（`~`）中，查找文件属性修改时间在 30 天内的文件，并限制查找深度为 3。
   - **用途**：查找最近修改过属性的文件，并限制搜索层级。
8. **find . -maxdepth 5 -uid 2023**
   - **含义**：在当前目录下，查找属于用户 ID 为 2023 的文件，最大查找深度为 5。
   - **用途**：查找属于特定用户的文件，并限制搜索深度。
9. **find . -maxdepth 3 -uid +999 -uid -1100 -exec ls -l {} \;**
   - **含义**：在当前目录下，查找用户 ID 大于 999 且小于 1100 的文件，并显示这些文件的详细信息（使用 `ls -l`）。最大查找深度为 3。
   - **用途**：查找属于特定用户 ID 范围的文件，并列出其详细信息。

`-exec` 与 `-ok` 参数说明：

- **-exec**：执行指定的命令，命令会对 `find` 查找到的每个文件执行一次。例如，`find ... -exec rm {} \;` 会删除查找到的每个文件。
- **-ok**：与 `-exec` 类似，但每次执行命令之前会询问用户是否确认执行。这对于防止意外删除或操作文件很有用。`-exec` 和 `-ok` 不能同时使用。

示例：

- `-exec rm {} \;`：会删除查找到的每个文件，不进行询问。
- `-ok rm {} \;`：会删除查找到的每个文件，但在执行每个删除操作时都会询问用户是否确认。

总结：

`find` 命令在 Linux/Unix 系统中非常有用，允许用户根据文件名、权限、大小、时间、用户等多种条件查找文件，并可以对查找到的文件进行操作（如删除、修改、列出详细信息等）。通过结合不同的参数，`find` 可以高效地完成各种复杂的文件搜索和管理任务。

注意：命令find logs -type f -mtime

+5 -exec -ok rm {} \;是错误的，-ok和-exec不能同时出现

11.P41:tar命令的用法，参考表2-17中的示例。

掌握参数c, t, x, z, j以及必带的vf（v表示显示输出，f表示文件），组合在一起就成了cvf, tvf, czvf, xzvf, cjvf, xjvf等。（单选）

常用参数：

1. **c**：创建新的归档文件（create）。
2. **t**：列出归档文件中的内容（list）。
3. **x**：解压归档文件（extract）。
4. **v**：显示详细输出（verbose），列出处理的文件。
5. **f**：指定归档文件的名称（file）。
6. **z**：通过 `gzip` 压缩归档文件，生成 `.tar.gz` 或 `.tgz` 文件。
7. **j**：通过 `bzip2` 压缩归档文件，生成 `.tar.bz2` 文件。

参数组合：

- **cvf**：创建归档文件并指定文件名，显示详细输出。
  - **例子**：`tar cvf archive.tar directory/` 将 `directory/` 目录打包成 `archive.tar` 文件，并显示详细输出。
- **tvf**：列出归档文件中的内容，显示详细信息。
  - **例子**：`tar tvf archive.tar` 列出 `archive.tar` 文件中的内容。
- **xzvf**：解压缩并提取 `.tar.gz` 格式的文件，同时显示详细输出。
  - **例子**：`tar xzvf archive.tar.gz` 解压并显示 `archive.tar.gz` 中的文件。
- **czvf**：创建 `.tar.gz` 压缩文件。
  - **例子**：`tar czvf archive.tar.gz directory/` 将 `directory/` 目录压缩为 `archive.tar.gz` 文件。
- **cjvf**：创建 `.tar.bz2` 压缩文件。
  - **例子**：`tar cjvf archive.tar.bz2 directory/` 将 `directory/` 目录压缩为 `archive.tar.bz2` 文件。
- **xjvf**：解压并提取 `.tar.bz2` 格式的文件。
  - **例子**：`tar xjvf archive.tar.bz2` 解压并显示 `archive.tar.bz2` 中的文件。

12.P43-46：vi/vim编辑器命令。（单选）

光标移动相关命令：

1. **3+**：

- **用途**：光标下移3行。
- **解释**：光标移动到当前行的下方3行。

2. **25-**：
   - **用途**：光标上移25行。
   - **解释**：光标移动到当前行的上方25行。
3. **:15**：
   - **用途**：将光标移动到第15行。
   - **解释**：光标直接跳转到第15行。

删除相关命令：

1. **dd**：
   - **用途**：删除光标所在行。
   - **解释**：删除当前光标所在的整行。
2. **2dw**：
   - **用途**：删除光标后2个词。
   - **解释**：删除从当前光标位置起，向后删除2个词。
3. **2d$**：
   - **用途**：从光标处开始删除2行。
   - **解释**：删除从当前光标所在行开始的2行。
4. **5dd**：
   - **用途**：从光标所在行开始往下删除5整行。
   - **解释**：删除当前行及下面的4行，一共5行。
5. **d$**：
   - **用途**：从光标处开始删除至行尾。
   - **解释**：删除从光标位置到当前行末尾的所有内容。

复制相关命令：

1. **yy**：
   - **用途**：复制光标所在行。
   - **解释**：复制当前光标所在的整行。
2. **5yy**：
   - **用途**：从光标处开始往后复制5行。
   - **解释**：复制当前行以及下面的4行，一共5行。

撤销与重做：

1. **u**：
   - **用途**：撤销上一步操作。
   - **解释**：撤销前一个编辑操作。
2. **+r**：
   - **用途**：重做操作。

- **解释**：重做上一步撤销的操作。

块操作：

1. `v`

   ：

   - **用途**：进入块操作模式，光标移动时可以选中字符块。
   - **解释**：按下 `v` 键进入可视模式，可以选中字符块，进行复制、剪切等操作。

剪切、复制与粘贴：

1. **c**：
   - **用途**：剪切选中的内容（在块操作模式下）。
   - **解释**：剪切选中的字符块。
2. **y**：
   - **用途**：复制选中的内容（在块操作模式下）。
   - **解释**：复制选中的字符块。
3. **p**：
   - **用途**：粘贴内容。
   - **解释**：粘贴剪切或复制的内容，通常粘贴在光标后的位置。

13. P54：输入、输出和错误重定向。>, >>, 2>, 2>>, <, &>符号的含义。（单选）

1. >：

- **用途**：将命令的标准输出（stdout）重定向到文件中，覆盖文件内容。
- **例子**：`echo "Hello, World!" > output.txt` 将 `"Hello, World!"` 输出到 `output.txt` 文件中，若文件存在，则覆盖内容。

2. >>：

- **用途**：将命令的标准输出（stdout）追加到文件的末尾。
- **例子**：`echo "Hello again!" >> output.txt` 会将 `"Hello again!"` 追加到 `output.txt` 文件的末尾，而不会覆盖原文件内容。

3. 2>：

- **用途**：将命令的标准错误输出（stderr）重定向到文件中，覆盖文件内容。
- **例子**：`ls non_existent_file 2> error.txt` 会将错误信息输出到 `error.txt` 文件中，若文件已存在，则覆盖内容。

4. 2>>：

- **用途**：将命令的标准错误输出（stderr）追加到文件的末尾。
- **例子**：`ls non_existent_file 2>> error.txt` 会将错误信息追加到 `error.txt` 文件的末尾。

5. <：

- **用途**：将文件的内容作为命令的标准输入（stdin）。
- **例子**：`sort < input.txt` 会将 `input.txt` 文件的内容作为输入传递给 `sort` 命令。

6. &>:

- **用途**：将标准输出（stdout）和标准错误输出（stderr）都重定向到同一个文件，覆盖文件内容。
- **例子**：`command &> output.txt` 会将 `command` 命令的标准输出和错误输出都重定向到 `output.txt` 文件中，若文件已存在，则覆盖内容。

总结:

- **>**：将标准输出重定向到文件，覆盖文件内容。
- **>>**：将标准输出追加到文件末尾。
- **2>**：将标准错误输出重定向到文件，覆盖文件内容。
- **2>>**：将标准错误输出追加到文件末尾。
- **<**：将文件内容作为命令的输入。
- **&>**：将标准输出和标准错误输出都重定向到文件，覆盖文件内容

14.P56：命令组合语法。;, &&, ||的语法。（单选）

1.；（分号）：

- **语法**：`c1; c2; c3`
- **解释**：命令按顺序依次执行，不论前一个命令的执行结果如何。如果前面的命令执行出错，后面的命令仍然会继续执行。
- 示例

  ：

  ```
  echo "First"; echo "Second"; echo "Third"
  ```

  - 上述命令会依次执行三个 `echo` 命令，不管前一个命令是否成功。

2. &&（与）：

- **语法**：`c1 && c2`
- **解释**：只有在 `c1` 执行成功（返回值为0）时，才会执行 `c2`。如果 `c1` 执行失败（返回值非0），`c2` 不会被执行。
- 示例

  ：

  ```
  mkdir test && cd test
  ```

  - 只有在 `mkdir test` 创建目录成功后，才会执行 `cd test` 命令进入该目录。如果 `mkdir` 失败，则 `cd test` 不会执行。

3. ||（或）：

- **语法**：`c1 || c2`
- **解释**：只有在 `c1` 执行失败（返回值非0）时，才会执行 `c2`。如果 `c1` 执行成功（返回值为0），`c2` 不会被执行。
- 示例

  ：

  ```
  mkdir test || echo "Directory already exists"
  ```

  - 如果 `mkdir test` 执行失败（比如目录已经存在），则输出 `"Directory already exists"`，否则不会执行。

4. 混合使用 && 和 ||：

- **语法**：`c1 && c2 || c3`
- **解释**：如果 `c1` 执行成功，则执行 `c2`，否则执行 `c3`。
- 示例

  ：

  ```
  echo "Test" && mkdir test || echo "Failed"
  ```

  - 如果 `echo "Test"` 成功，接着执行 `mkdir test`，如果 `mkdir` 失败，执行 `echo "Failed"`。

5. ！（非）：

- **语法**：`! c1 && c2`
- **解释**：`!` 表示取反，即如果 `c1` 执行失败（返回值非0），则执行 `c2`。
- 示例

  ：

  ```
  ! mkdir test && echo "Directory already exists"
  ```

  - 如果 `mkdir test` 失败（即目录已经存在），执行 `echo "Directory already exists"`。

总结：

- **;**：无论前一个命令是否成功，都会执行下一个命令。
- **&&**：前一个命令成功时才执行下一个命令。
- **||**：前一个命令失败时才执行下一个命令。
- **!**：取反，用来判断前一个命令是否失败，失败则执行后续命令。

15.P60：账户系统文件。/etc/passwd, /etc/shadow, /etc/group, /etc/gshadow存储内容。（单选）

在 Linux 系统中，账户相关的系统文件存储着与用户和组相关的信息。以下是几个重要账户系统文件的内容和作用：

1. /etc/passwd：

- **存储内容**：存储系统中所有用户的基本信息，包括用户名、密码占位符、UID、GID、用户全名、家目录、默认 shell 等。
- **格式**：每行代表一个用户，字段之间用冒号（`:`）分隔。
- 示例

  ：

  ```
  username:x:1000:1000:User Name:/home/username:/bin/bash
  ```

  - 字段解释

    ：

    1. **用户名**：用户的登录名。
    2. **密码占位符**（通常为 `x`）：指向 `/etc/shadow` 文件中的加密密码。
    3. **UID**：用户的唯一标识符（User ID）。
    4. **GID**：用户主组的唯一标识符（Group ID）。
    5. **用户全名**：用户的描述信息（通常为实际姓名）。
    6. **家目录**：用户的主目录路径。
    7. **默认 shell**：用户登录后的默认 shell 程序。

2. /etc/shadow：

- **存储内容**：存储用户的加密密码和密码相关的配置信息，如密码过期日期、账户失效日期等。该文件只能由 root 用户访问。
- **格式**：每行代表一个用户，字段之间用冒号（`:`）分隔。
- 示例

  ：

  ```
  username:$6$randomsalt$encryptedpassword:18000:0:99999:7:::
  ```

  - 字段解释

    ：

    1. **用户名**：用户的登录名。
    2. **加密密码**：存储加密的密码，通常使用 `crypt` 函数。
    3. **密码最近更改的天数**：距离 Unix 纪元（1970年1月1日）以来的天数。
    4. **最小密码年龄**：用户修改密码的最小天数限制。
    5. **最大密码年龄**：用户密码的最大使用期限。
    6. **警告天数**：密码到期前的警告天数。
    7. **禁用天数**：账户被禁用的日期。
    8. **账户过期日期**：账户过期的日期。
    9. **保留字段**：保留字段，不常用。

3. /etc/group：

- **存储内容**：存储系统中所有用户组的信息，包括组名、组密码占位符、GID 和成员列表。
- **格式**：每行代表一个组，字段之间用冒号（`:`）分隔。
- 示例

  ：

  ```
  groupname:x:1000:user1,user2
  ```

  - 字段解释

    ：

    1. **组名**：用户组的名称。
    2. **组密码占位符**（通常为 `x`）：历史上曾用于存储组密码，现在通常为空。
    3. **GID**：组的唯一标识符（Group ID）。
    4. **组成员**：属于该组的用户列表。

4. /etc/gshadow：

- **存储内容**：存储组的加密密码和与组相关的安全信息。类似于 `/etc/shadow`，但它针对组，且只能由 root 用户访问。
- **格式**：每行代表一个组，字段之间用冒号（`:`）分隔。
- 示例

  ：

  ```
  groupname:!::user1,user2
  ```

  - 字段解释

    ：

    1. **组名**：用户组的名称。
    2. **组密码**：组的加密密码（如果有）。如果没有密码或禁用密码，通常为 `!`。
    3. **组管理员**：可以管理该组的用户列表。
    4. **组成员**：属于该组的用户列表。

总结：

- **/etc/passwd**：存储所有用户的基本信息，如用户名、UID、GID、家目录等。
- **/etc/shadow**：存储用户的加密密码和密码策略。
- **/etc/group**：存储所有用户组的基本信息，如组名、GID 和成员。
- **/etc/gshadow**：存储组的加密密码和组的安全策略。

16.P63：账户管理命令。useradd, usermod, userdel, groupadd, groupmod, groupdel，id，newgrp命令的基本用法。（单选，简答）

账户管理命令的基本用法：

1. useradd：

- **用途**：创建一个新的用户。
- 常用参数
  ：
  - `-d`：指定用户的主目录。
  - `-g`：指定用户的主组。
  - `-G`：指定附加组（多个附加组用逗号分隔）。
- 示例
  ：

```
useradd -g mario u1  # 创建用户u1，主组为mario
passwd u1            # 为用户u1设置密码
```

2. usermod：

- **用途**：修改现有用户的信息。
- 常用参数
  ：
  - `-l`：修改用户名。
  - `-G`：指定用户的附加组。
- 示例
  ：

```
usermod -l user1 u1  # 将用户u1改名为user1
usermod -G g1 mario  # 为用户mario指定附加组g1
```

3. userdel：

- **用途**：删除用户。
- 常用参数
  ：
  - `-r`：在删除用户的同时删除该用户的主目录。
- 示例
  ：

```
userdel -r user1  # 删除用户user1，并删除其主目录
```

4. groupadd：

- **用途**：创建一个新的用户组。

- 示例

  ：

```
groupadd g1  # 创建组g1
```

5. groupmod：

- **用途**：修改现有用户组的信息。
- 常用参数

  ：

  - `-n`：修改组名。
- 示例

  ：

```
groupmod -n group1 g1  # 将组g1改名为group1
```

6. groupdel：

- **用途**：删除一个现有用户组。
- 示例

  ：

```
groupdel g1  # 删除组g1
```

7. id：

- **用途**：查看用户的属性，如 UID、GID、附加组等。
- 示例

  ：

```
id mario  # 查看用户mario的属性
```

8. newgrp：

- **用途**：切换到指定的用户组。
- 示例

  ：

```
newgrp group1  # 切换到组group1
```

9. su：

- **用途**：切换到另一个用户的身份（需要密码）。

- 示例

  ：

```
su - mario  # 切换到用户mario
```

示例命令（按要求完成）：

1. `useradd -g mario u1` # 创建用户u1，主组为mario
2. `passwd u1` # 为用户u1设置密码
3. `usermod -l user1 u1` # 将用户u1改名为user1
4. `userdel -r user1` # 删除用户user1，并删除其主目录
5. `groupadd g1` # 创建组g1
6. `usermod -G g1 mario` # 为用户mario指定附加组g1
7. `id mario` # 查看用户mario的属性
8. `groupmod -n group1 g1` # 将组g1改名为group1
9. `su - mario` # 切换到用户mario
10. `newgrp group1` # 切换到组group1

---

17. P70：chmod命令的语法。符号ugo+-=，以及等价的数值设定法如644。（单选）

在 Linux 中，`chmod` 命令用于改变文件或目录的权限。该命令可以使用符号法（`ugo+-=`）或数值法（例如 `644`）来设置权限。

1. 符号法 (ugo+-=):

- **u**：表示文件的 **所有者**（User）。
- **g**：表示文件的 **所属组**（Group）。
- **o**：表示 **其他用户**（Others）。
- **a**：表示所有用户（all），等同于 `u+g+o`。

操作符：

- **+**：表示 **添加** 权限。
- **-**：表示 **删除** 权限。
- **=**：表示 **设置** 权限，覆盖原有的权限。

权限：

- **r**：读取权限（read）。
- **w**：写入权限（write）。
- **x**：执行权限（execute）。

常用示例：

- `chmod u+x file`：为文件的所有者添加执行权限。
- `chmod g-w file`：从所属组中删除写入权限。
- `chmod o=r file`：将其他用户的权限设置为只读。

2. 数值法：

数值法通过三个数字来设置文件权限，每个数字对应 **用户、组、其他用户** 的权限。这些数字分别代表：

- **4**：读取权限（`r`）。
- **2**：写入权限（`w`）。
- **1**：执行权限（`x`）。

权限的组合由这三个数字相加组成：

- **7**：读、写、执行（`rwx`），即 `4+2+1`。
- **6**：读、写（`rw-`），即 `4+2`。
- **5**：读、执行（`r-x`），即 `4+1`。
- **4**：读（`r--`），即 `4`。
- **3**：写、执行（`wx-`），即 `2+1`。
- **2**：写（`w--`），即 `2`。
- **1**：执行（`x--`），即 `1`。

常见权限设置：

- `chmod 644 file`：所有者有读写权限，组和其他用户只有读取权限（`rw-r--r--`）。
- `chmod 755 file`：所有者有读写执行权限，组和其他用户只有读取执行权限（`rwxr-xr-x`）。
- `chmod 777 file`：所有用户都有读写执行权限（`rwxrwxrwx`）。

示例：

1. **符号法**：
   - `chmod u+x file` # 给文件的所有者添加执行权限。
   - `chmod g-w file` # 给所属组删除写入权限。
   - `chmod o=r file` # 给其他用户设置只读权限。
2. **数值法**：
   - `chmod 644 file` # 所有者有读写权限，组和其他用户只有读取权限。
   - `chmod 755 file` # 所有者有读写执行权限，组和其他用户只有读取和执行权限。

总结：

- **符号法**：使用 `u`，`g`，`o` 来指定用户、组和其他用户的权限，通过 `+`，`-`，`=` 来添加、删除或设置权限。
- **数值法**：使用数字 `0-7` 来表示不同的权限组合。

18.P73：文件和文件夹的特殊权限。文件上的suid和sgid权限的作用。文件夹上sgid和sticky-bit（防删除位）的作用。（单选）

在 Linux 中，文件和目录的特殊权限有 **SUID**、**SGID** 和 **Sticky bit**，它们的作用如下：

1. SUID (Set User ID)：

- **作用**：当文件设置了 SUID 权限时，执行该文件的用户将临时获取该文件所有者的权限，而不是执行者本人的权限。通常用于需要特权权限的程序，比如 `passwd`。
- **表示**：`rws` 或 `r-s`，其中的 `s` 表示 SUID 权限。

- 示例

  ：

  ```
  -rwsr-xr-x 1 root root 12345 Oct  1 12:00 /usr/bin/passwd
  ```

  在这个示例中，

  ```
  passwd
  ```

  程序有 SUID 权限，意味着普通用户执行

  ```
  passwd
  ```

  命令时，会使用

  ```
  root
  ```

  用户的权限。

2. SGID (Set Group ID)：

- 作用

  ：

  - **在文件上**：当一个文件设置了 SGID 权限时，执行该文件的用户将临时获得该文件所属组的权限，而不是执行者当前组的权限。
  - **在目录上**：当一个目录设置了 SGID 权限时，所有在该目录下创建的文件或子目录将继承该目录的组，而不管创建者所属的组是什么。
- **表示**：`rwxs` 或 `rwxs`，其中的 `s` 表示 SGID 权限。
- 示例

  ：

  - 文件上的 SGID

    ：

    ```
    -rwxr-sr-x 1 root staff 12345 Oct  1 12:00 /usr/bin/somefile
    ```

    执行该文件时，进程将继承文件所属组（在此例中为

    ```
    staff
    ```

    ）的权限。
  - 目录上的 SGID

    ：

```
drwxrwsr-x 2 root staff 4096 Oct  1 12:00 /home/shared
```

在这个目录下创建的文件将继承

```
staff
```

组，即使创建者不属于该组。

3. Sticky Bit (防删除位):

- **作用**：设置 Sticky Bit 后，只有文件的所有者、目录的所有者或 root 用户才可以删除该目录中的文件。其他用户不能删除其他用户的文件。通常应用于公共可写目录，如 `/tmp`。
- **表示**：`rwxrwxrwt`，其中的 `t` 表示 Sticky Bit 权限。
- 示例

  ：

  ```
  drwxrwxrwt  9 root root 4096 Dec  1 14:00 /tmp
  ```

  在这个示例中，

  ```
  /tmp
  ```

  目录有 Sticky Bit 权限，意味着即使普通用户可以写入该目录，也无法删除其他用户的文件。

总结:

- **SUID**：在文件上设置 SUID 后，执行该文件的用户将继承该文件所有者的权限。
- SGID

  ：

  - 在文件上：执行该文件的用户将继承文件所属组的权限。
  - 在目录上：创建的新文件将继承目录的组。
- **Sticky Bit**：在目录上设置 Sticky Bit 后，只有文件的所有者、目录的所有者或 root 用户可以删除目录中的文件。

19.P75：文件ACL权限（单选，简答）

文件 ACL 权限 (Access Control Lists)

ACL（访问控制列表）是 Linux 系统中用来为文件或目录设置更细粒度的访问控制的机制。传统的文件权限模型（通过 `rwx` 和三种用户类型：所有者、组、其他）只能设置有限的权限，而 ACL 允许为特定用户、特定组设置不同的权限。

ACL 的常见命令：

1. **setfacl**：用于设置文件或目录的 ACL 权限。
   - `setfacl -m u:u1:rw f3`：为用户 `u1` 设置文件 `f3` 的读写权限。
   - `setfacl -m g:g1:rx f3`：为组 `g1` 设置文件 `f3` 的读取和执行权限。
   - `setfacl -m m:r f3`：设置文件 `f3` 的掩码（mask），即限制非所有者用户和组的最大权限为读取权限。
   - `setfacl -m d:u:u1:rw /root`：为目录 `/root` 设置默认 ACL 权限，新建的文件会自动继承该权限。
2. **getfacl**：用于查看文件或目录的 ACL 权限。
   - `getfacl f3`：查看文件 `f3` 的 ACL 权限。
3. **setfacl -x**：用于删除文件或目录中某个用户或组的 ACL 权限。
   - `setfacl -x u:u1 f3`：删除用户 `u1` 对文件 `f3` 的权限。
   - `setfacl -x g:g1 f3`：删除组 `g1` 对文件 `f3` 的权限。

ACL 权限表示：

- 如果文件或目录具有 ACL 属性，则其文件信息的第 11 个字符会从

  ```
  .
  ```

  变为

  ```
  +
  ```

  。
  - 示例

    ：

    ```
    -rw-rw-r--+ 1 root root 6 12月 18 11:09 f3
    -rw-r--r--. 1 root root 7 12月 18 11:10 f4
    ```


权限示例：

1. **setfacl -m u:u1:rw f3**：
   - 该命令将文件 `f3` 的 ACL 权限设置为用户 `u1` 拥有读写权限。
2. **setfacl -m g:g1:rx f3**：
   - 该命令将文件 `f3` 的 ACL 权限设置为组 `g1` 拥有读取和执行权限。
3. **setfacl -m m:r f3**：
   - 该命令设置文件 `f3` 的掩码（mask），使除文件所有者外，其他用户和组的最大权限为读取权限（`r`）。
4. **setfacl -m d:u:u1:rw /root**：
   - 该命令为目录 `/root` 设置默认权限，任何新建的文件都会自动拥有用户 `u1` 的读写权限。

5. **setfacl -x u:u1 f3**：
   - 该命令删除用户 `u1` 对文件 `f3` 的权限。
6. **setfacl -x g:g1 f3**：
   - 该命令删除组 `g1` 对文件 `f3` 的权限。

总结：

- **ACL** 使得 Linux 文件权限更加灵活，可以为不同的用户和组设置不同的权限。
- 使用 **setfacl** 设置 ACL 权限，使用 **getfacl** 查看文件的 ACL 权限。
- **掩码（mask）** 是一个重要的概念，它限制了组和其他用户的最大权限。

20.P83：作业控制命令和快捷键。命令结尾加&符号，jobs, fg, bg命令和快捷键Ctrl+Z的用途。（单选）

作业控制命令和快捷键

在 Linux 中，作业控制是指对运行在后台或前台的进程进行管理。常用的作业控制命令包括 `jobs`、`fg`、`bg`，以及快捷键 `Ctrl+Z` 和在命令行中使用 `&` 符号来将命令放入后台执行。

常用命令与快捷键用途：

1. **命令结尾加 & 符号**：
   - **用途**：将命令放到后台执行。加上 `&` 后，命令会在后台运行，不会占用当前的终端，可以继续执行其他命令。
   - 示例

     ：

     ```
     long_running_command &
     ```

     该命令将在后台执行，且终端立即返回提示符。
2. **jobs 命令**：
   - **用途**：列出当前会话中所有的作业（后台进程）。输出会显示作业编号以及该作业的状态（如正在运行、已停止等）。
   - 示例

     ：

     ```
     jobs
     ```

     输出可能类似：

     ```
     [1]+  1234 Running                 long_running_command &
     [2]-  1235 Stopped                 another_command
     ```
3. **fg 命令**：

- **用途**：将后台作业带到前台执行。你可以通过作业编号来指定要恢复的作业。
- 示例

  ：

  ```
  fg %1
  ```

  该命令会将作业编号为 1 的后台作业带到前台执行。

4. **bg 命令**：
   - **用途**：将已停止的作业放到后台继续执行。你可以通过作业编号来指定要恢复的作业。
   - 示例

     ：

     ```
     bg %1
     ```

     该命令会将作业编号为 1 的停止作业放到后台继续执行。
5. **Ctrl+Z 快捷键**：
   - **用途**：将当前正在前台运行的进程挂起（暂停）并将其放到后台。执行该快捷键后，可以使用 `bg` 命令将进程继续在后台运行，或使用 `fg` 命令将进程带回前台。
   - 示例

     ：

     - 假设你正在运行一个程序，按下 `Ctrl+Z` 后，程序会被挂起，终端会显示该进程的作业编号。
     - 之后，你可以使用 `bg` 将其放到后台，或者使用 `fg` 将其恢复到前台。

总结：

- **&**：将命令放到后台执行。
- **jobs**：列出当前会话中的作业。
- **fg**：将后台作业带到前台。
- **bg**：将已停止的作业放到后台继续执行。
- **Ctrl+Z**：挂起当前前台进程并放到后台。

21.P96

LVM管理示例

① 将三块硬盘初始化为物理卷

```
pvcreate /dev/sdb /dev/sdc /dev/sdd
```

- `pvcreate` 命令将物理设备 `/dev/sdb`，`/dev/sdc`，`/dev/sdd` 初始化为物理卷（Physical Volume）。

② 新建一个卷组httpd，并将以上三块硬盘加入该卷组

```
vgcreate httpd /dev/sdb /dev/sdc /dev/sdd
```

- `vgcreate` 命令新建一个卷组（Volume Group）`httpd`，并将三个物理卷 `/dev/sdb`，`/dev/sdc`，`/dev/sdd` 加入到该卷组。

③ 在卷组httpd上新建一个逻辑卷www1，大小为2TB

```
lvcreate -n www1 -L 2T httpd
```

- `lvcreate` 命令在卷组 `httpd` 上创建一个逻辑卷（Logical Volume）`www1`，大小为 2TB。

④ 将逻辑卷格式化为ext3文件系统

```
mkfs.ext3 /dev/httpd/www1
```

- `mkfs.ext3` 命令将逻辑卷 `/dev/httpd/www1` 格式化为 `ext3` 文件系统。

⑤ 新建一个文件夹/var/www1，并将逻辑卷www1挂载到该文件夹上

```
mkdir /var/www1
mount /dev/httpd/www1 /var/www1
```

- `mkdir` 命令创建 `/var/www1` 目录。
- `mount` 命令将逻辑卷 `/dev/httpd/www1` 挂载到 `/var/www1` 目录。

⑥ 扩展逻辑卷www1，容量增大1TB

```
lvextend -L +1T /dev/httpd/www1
```

- `lvextend` 命令将逻辑卷 `/dev/httpd/www1` 扩展 1TB。

⑦ 将逻辑卷容量的变动告诉文件系统，让文件系统做对应修改

```
resize2fs /dev/httpd/www1
```

- `resize2fs` 命令使文件系统调整大小，以适应逻辑卷的扩展。

⑧ 给逻辑卷/dev/httpd/www1创建一个快照，名叫www1s，大小为0.5GB，用于不停止逻辑卷/dev/httpd/www1操作的情况下，备份快照

```
lvcreate -s -n www1s -L 0.5G /dev/httpd/www1
```

- `lvcreate` 命令创建一个名为 `www1s` 的快照，大小为 0.5GB，用于备份。

RAID管理示例

① 将硬盘sdb、sdc、sdd和sde用mdadm命令配置为RAID5级别，名称为/dev/md1，其中sde为备用盘

```
mdadm -C /dev/md1 -l5 -n3 /dev/sdb /dev/sdc /dev/sdd -x1 /dev/sde
```

- `mdadm` 命令创建一个 RAID5 设备 `/dev/md1`，使用硬盘 `/dev/sdb`，`/dev/sdc`，`/dev/sdd`，并指定 `/dev/sde` 作为备用盘。

② 将md1格式化为ext3文件系统

```
mkfs.ext3 /dev/md1
```

- `mkfs.ext3` 命令将 RAID 设备 `/dev/md1` 格式化为 `ext3` 文件系统。

③ 新建一个文件夹/var/ftp1，并将md1挂载到该文件夹上

```
mkdir /var/ftp1
mount /dev/md1 /var/ftp1
```

- `mkdir` 命令创建 `/var/ftp1` 目录。
- `mount` 命令将 RAID 设备 `/dev/md1` 挂载到 `/var/ftp1` 目录。

④ 保存配置到/etc/mdadm.conf中

```
mdadm -D -s >> /etc/mdadm.conf
```

- `mdadm -D -s` 命令扫描 RAID 配置信息，并将其保存到 `/etc/mdadm.conf` 配置文件中。

⑤ 将md1卸载

```
umount /dev/md1
```

- `umount` 命令卸载 RAID 设备 `/dev/md1`。

⑥ 将RAID设备md1停止

```
mdadm -S /dev/md1
```

- `mdadm -S` 命令停止 RAID 设备 `/dev/md1`。

⑦ 将磁盘/dev/sdb数据清空

```
mdadm --zero-superblock /dev/sdb
```

- `mdadm --zero-superblock` 命令清除磁盘 `/dev/sdb` 的超级块（superblock），即清除磁盘上的 RAID 配置信息。

总结

- **LVM管理**：通过 `pvcreate`，`vgcreate`，`lvcreate` 等命令，您可以创建和管理物理卷、卷组、逻辑卷等，完成文件系统格式化、挂载、扩展等操作。
- **RAID管理**：通过 `mdadm` 命令配置 RAID 设备，支持创建、格式化、挂载、停止等操作，可以配置不同的 RAID 级别（如 RAID5），并且能够进行磁盘清空等管理。

22.P113：一致的网络设备命名规则（单选）。

在 Linux 系统中，一致的网络设备命名规则（Predictable Network Interface Names）用于为网络设备分配有意义的名称。这些规则为每个网络接口分配一个名称，避免了传统的 `eth0`、`eth1` 等名称的随机性。以下是各个前缀的含义：

1. **en**: 代表 **Ethernet** 网络接口（有线网卡）。通常与以太网（Ethernet）接口相关联。
   - 示例: `enp0s3` 代表一个以太网接口，位于第 0 个 PCI 插槽的第 3 个设备。
2. **wl**: 代表 **Wireless LAN** 网络接口（无线网卡）。通常与无线网络适配器相关联。
   - 示例: `wlp2s0` 代表一个无线网卡，位于第 2 个 PCI 插槽的第 0 个设备。
3. **ww**: 代表 **WWAN**（Wireless Wide Area Network，无线广域网）设备，通常指的是移动数据卡或其他广域网设备。
   - 示例: `wwp0s20u1` 可能是一个广域网设备，连接到 USB 插口。
4. **o**: 代表 **Other** 类型的设备，用于其他不常见的网络接口设备。
5. **s**: 代表 **Serial** 网络接口，通常与串行设备相关。
6. **p**: 代表 **PCI** 网络接口，表示设备是通过 PCI 总线连接的。

这种一致的命名方式是为了确保网络接口名称的一致性和可靠性，尤其是在具有多个网络接口的服务器和虚拟化环境中，可以有效避免设备名称的混乱。

例子

- `enp0s3`: Ethernet 设备，PCI 设备位于总线 0 的插槽 3。
- `wlp2s0`: Wireless LAN 设备，PCI 设备位于总线 2 的插槽 0。

23.P114：ip命令（临时配置网络参数，重启后失效。单选，简答）

`ip` 命令是用于临时配置网络参数的工具，常用于查看和配置网络接口、IP 地址、路由等。以下是常见用法：

1. **显示全部接口的IP地址：**

   ```
   ip a
   ```

   该命令显示所有网络接口的IP地址、MAC地址、网络状态等信息。

2. **临时增加一个IP地址：**

   ```
   ip addr add 192.168.0.1/24 dev ens33
   ```

   该命令会为名为 `ens33` 的网络接口临时添加一个 IP 地址 `192.168.0.1/24`。这项更改在重启后会失效。

3. **显示路由表：**

   ```
   ip r
   ```

   该命令显示当前的路由表。

4. **临时增加一个路由：**

```
ip r add 192.168.10.0/24 via 192.168.0.2 dev ens33
```

该命令会为目标网络 `192.168.10.0/24` 添加一个路由，数据包通过网关 `192.168.0.2` 转发，并通过接口 `ens33` 发送。这个路由也是临时的，重启后失效。

这些 `ip` 命令配置的网络参数是临时的，也就是说，在系统重启后，它们会丢失。如果需要使配置永久生效，可以将相关配置添加到系统的网络配置文件中（如 `/etc/network/interfaces` 或 `/etc/sysconfig/network-scripts/ifcfg-*`）。

24.P117：/etc/hosts文件的功能和格式

`/etc/hosts` 文件是 Linux 系统中的一个本地 DNS 解析文件，用于将主机名映射到IP地址。它提供了一种方式来在没有 DNS 服务器的情况下解析主机名。该文件的作用是，在网络通信时，当程序或用户需要通过主机名进行通信时，系统会首先查询该文件，将主机名解析为对应的 IP 地址。

功能：

- **本地域名解析：** 当访问某个主机名时，系统会首先查找 `/etc/hosts` 文件，如果文件中有该主机名对应的 IP 地址，则直接使用该 IP 地址。
- **提供网络服务：** 可以为本地开发环境、测试环境或内部网络的设备配置别名，而不需要依赖外部 DNS 服务。
- **设置主机名与IP的映射：** 用于将主机名与 IP 地址绑定，确保在没有网络的情况下，系统可以正常解析本地主机名。

格式：

`/etc/hosts` 文件的格式由多行组成，每一行包含以下信息：

1. **IP 地址**：指定主机的 IP 地址。
2. **主机名**：主机的名字（包括完全限定域名 FQDN 和主机别名）。
3. **可选的别名**：可以为主机名指定多个别名，用空格或制表符分隔。

格式示例：

```
127.0.0.1   localhost
192.168.1.10   example.com example
192.168.1.20   webserver.local webserver
```

详细说明：

- **127.0.0.1**：这是本地回环地址（localhost），代表计算机自身。
- **localhost**：是本地回环接口的主机名，通常与 IP 地址 `127.0.0.1` 配对。
- **192.168.1.10**：是内网设备的 IP 地址。
- **example.com**：是设备的完全限定域名（FQDN），即设备的正式名称。
- **example**：是设备的别名。

其他说明：

- 每行的注释以 `#` 开头，注释内容会被系统忽略。

- 每个主机名与 IP 地址之间必须使用至少一个空格或制表符分隔。

示例：

```
# 本地回环接口
127.0.0.1   localhost

# 网络中其他设备
192.168.1.100   myserver.example.com myserver
```

注意：

1. `/etc/hosts` 文件的内容是静态的，在文件中做的修改需要保存后才能生效。
2. 该文件的优先级较高，在进行域名解析时，系统会首先查询 `/etc/hosts` 文件，再查询 DNS 服务器。

25.P117：/etc/resolv.conf文件的功能和格式

`/etc/resolv.conf` 文件是 Linux 系统中的一个配置文件，用于配置域名解析（DNS）。它告诉系统如何将域名转换为 IP 地址，通常由 DNS 服务器提供解析服务。该文件指定了 DNS 服务器的 IP 地址、搜索域名等信息。

功能：

- **配置 DNS 服务器：** `/etc/resolv.conf` 文件指定了系统使用的 DNS 服务器的 IP 地址。当需要进行域名解析时，系统会查询这些 DNS 服务器。
- **指定搜索域：** 文件中还可以设置搜索域（search domain），使得系统在解析域名时可以自动加上域后缀。
- **域名解析配置：** 通过该文件，系统能够知道如何查询 DNS 服务器以解析外部或内部的域名。

格式：

`/etc/resolv.conf` 文件的格式由几种不同的配置项组成，每一行以特定关键字开头，后面跟着相应的参数。

常见配置项：

1. **nameserver**：指定一个 DNS 服务器的 IP 地址。
2. **search**：指定域名搜索列表。当给定的主机名不能解析时，系统会自动加上这些域名后缀进行尝试。
3. **domain**：指定默认域名（与 `search` 类似，但 `domain` 只能指定一个域）。
4. **options**：提供额外的解析选项（如超时设置）。

格式示例：

```
# 指定 DNS 服务器
nameserver 8.8.8.8
nameserver 8.8.4.4

# 搜索域，系统会在主机名后自动添加此域名进行解析
search example.com

# 指定默认域名（可选）
domain example.com

# 设置额外的选项（例如：设置 DNS 查询超时时间）
options timeout:5
```

详细说明：

- **nameserver**：每个 `nameserver` 行后跟一个 IP 地址，用于指定 DNS 服务器。系统会按照顺序查询这些服务器。如果第一个服务器无法响应请求，则会查询第二个，依此类推。
- **search**：指定一个或多个域名后缀。如果一个主机名无法解析，系统会尝试在该主机名后附加这些域名。例如，如果配置了 `search example.com`，那么访问 `server1` 时，系统会自动尝试解析 `server1.example.com`。
- **domain**：通常与 `search` 配置类似，但 `domain` 只支持一个域名。它是系统的默认域名，影响某些程序的行为。
- **options**：用于指定 DNS 查询的一些参数，比如超时设置（`timeout`）或最大查询次数（`attempts`）。

示例：

```
# 使用 Google 的 DNS 服务器
nameserver 8.8.8.8
nameserver 8.8.4.4

# 搜索域
search mycompany.local

# 设置 DNS 查询超时为 3 秒
options timeout:3
```


其他配置：

- `options attempts:2`：指定 DNS 服务器尝试解析的次数，默认是 2 次。
- `options rotate`：启用轮询 DNS 服务器，即每次查询时轮流使用不同的 DNS 服务器。

注意：

1. `/etc/resolv.conf` 文件中的配置仅对系统中的应用程序（如浏览器、命令行工具）有效。
2. 如果系统使用了 DHCP（动态主机配置协议），`/etc/resolv.conf` 文件的内容通常由 DHCP 服务器提供。在这种情况下，可以通过修改 DHCP 客户端配置文件来控制 `/etc/resolv.conf` 文件的内容。
3. 在某些系统上，`/etc/resolv.conf` 文件可能是由 `systemd-resolved` 管理的，这意味着其内容可能会自动更新，不易手动修改。

26.P118：开启包转发功能：sysctl net.ipv4.ip_forward=1

开启包转发功能

在 Linux 系统中，包转发（IP forwarding）是指允许路由器将一个网络接口收到的 IP 数据包转发到另一个网络接口。开启包转发通常用于设置路由器或实现 NAT（网络地址转换）功能。

开启包转发：

```
sysctl net.ipv4.ip_forward=1
```

- 该命令临时启用包转发功能，将 `net.ipv4.ip_forward` 参数设置为 `1`，表示开启 IPv4 包转发。

关闭包转发：

```
sysctl net.ipv4.ip_forward=0
```

- 该命令临时禁用包转发功能，将 `net.ipv4.ip_forward` 参数设置为 `0`，表示关闭 IPv4 包转发。

使配置立即生效：

```
sysctl -p
```

- 执行 `sysctl -p` 命令会重新加载 `/etc/sysctl.conf` 配置文件中的所有设置，并立即生效。
- 如果没有执行 `sysctl -p`，配置只会在系统重启后生效。

配置文件永久生效：

如果希望包转发在系统重启后仍然生效，可以直接修改 `/etc/sysctl.conf` 文件。

1. 打开 `/etc/sysctl.conf` 文件：

   ```
   vi /etc/sysctl.conf
   ```

2. 在文件中添加或修改以下行：

   ```
   net.ipv4.ip_forward=1
   ```

3. 保存并退出后执行 `sysctl -p` 来使更改生效。

这样设置后，系统每次启动时都会自动启用 IP 包转发功能。

注意事项：

- 包转发通常用于路由器或具有多个网络接口的服务器，将数据包从一个网络转发到另一个网络。
- 如果你在 NAT 配置或设置 VPN 服务器等情景下使用包转发，它是必需的。 □

27.P120：nmcli命令（永久性配置网络参数，也就是写到配置文件里，但不立即生效，需立即生效要用下面两行命令关闭并启动网络接口。单选，简答）

`nmcli` 命令（用于永久性配置网络参数）

`nmcli` 是 NetworkManager 的命令行界面，可以用来配置和管理网络连接。与 `ip` 命令不同，`nmcli` 更加注重网络配置的持久性，即设置的参数会写入配置文件中。需要注意的是，使用 `nmcli` 设置网络参数时，这些更改并不会立即生效，除非手动重启网络接口。

常用 `nmcli` 命令：

1. **关闭网络接口 ens33：**

   ```
   nmcli d down ens33
   ```

   - 关闭 `ens33` 网络接口。
2. **启动网络接口 ens33：**

   ```
   nmcli c up ens33
   ```

   - 启动并连接 `ens33` 网络接口。
3. **查看所有网络接口的状态：**

   ```
   nmcli d
   ```

   - 列出系统中的所有网络接口，并显示它们的连接状态（已连接或断开）。
4. **设置 ens33 自动获取 IP 地址：**

   ```
   nmcli c m ens33 ipv4.method auto
   ```

   - 配置 `ens33` 接口为自动获取 IP 地址（DHCP）。
5. **手动设置 ens33 的 IP 地址：**

   ```
   nmcli c m ens33 ipv4.method manual ipv4.addresses 192.168.0.5/24
   ```

   - 设置 `ens33` 接口为手动配置 IP 地址，并指定地址为 `192.168.0.5/24`。
6. **为 ens33 添加一个新的 IP 地址：**

   ```
   nmcli c m ens33 +ipv4.addresses 10.0.0.5/24
   ```

   - 在 `ens33` 上增加一个新的 IP 地址 `10.0.0.5/24`。
7. **从 ens33 删除指定的 IP 地址：**

   ```
   nmcli c m ens33 -ipv4.addresses 192.168.1.5/24
   ```

   - 删除 `ens33` 接口上的 IP 地址 `192.168.1.5/24`。
8. **设置 ens33 的网关：**

```
nmcli c m ens33 ipv4.gateway 192.168.0.1
```

   - 设置 `ens33` 的默认网关为 `192.168.0.1`。

9. **设置 DNS 服务器：**

```
nmcli c m ens33 ipv4.dns 222.206.176.61
```

   - 设置 `ens33` 使用的 DNS 服务器为 `222.206.176.61`。

使更改生效：

`nmcli` 命令更改的网络配置会永久保存在配置文件中，但这些更改通常不会立即生效，除非你手动重启网络接口。

- 关闭并启动网络接口来使配置立即生效：

```
nmcli d down ens33  # 关闭接口
nmcli c up ens33    # 启动接口
```

这样，你可以使用 `nmcli` 对网络进行永久配置，并通过重新启动接口使配置生效。

28.P135：yum命令（单选）要求知道以下参数：

`yum` 命令的常见用法

`yum`（Yellowdog Updater, Modified）是基于RPM包管理系统的一个前端工具，广泛用于Red Hat及其衍生系统（如CentOS、Fedora）中来安装、卸载、更新软件包。以下是一些常用的 `yum` 命令及其参数的解释：

常见命令及参数：

1. **安装软件包：**

```
yum install <package-name>
```

   - 用于安装一个新的软件包。例如，要安装

```
wget
```

     软件包，可以执行：

```
yum install wget
```

2. **查看软件包信息：**

```
yum info <package-name>
```

   - 显示有关某个软件包的详细信息，例如版本、大小、依赖关系等。例如，查看

```
wget
```

软件包的信息：

```
yum info wget
```

3. **列出所有已安装的软件包：**

```
yum list installed
```

   - 显示系统上所有已安装的软件包。例如，查看所有已安装的软件包：

```
yum list installed
```

4. **列出所有可更新的软件包：**

```
yum list updates
```

   - 显示所有可以更新的软件包。例如，查看可以更新的软件包：

```
yum list updates
```

5. **卸载软件包：**

```
yum remove <package-name>
```

或者

```
yum erase <package-name>
```

   - 用于卸载已安装的软件包。例如，卸载

```
wget
```

软件包：

```
yum remove wget
```

或者

```
yum erase wget
```

总结：

- **install**: 安装新软件包。
- **info**: 查看软件包信息。

- **list installed**: 列出所有已安装的软件包。
- **list updates**: 列出所有可更新的软件包。
- **remove 或 erase**: 卸载软件包。

这些命令使得管理系统的软件包变得非常简便，并且能够快速获取软件包的状态、信息和进行更新。

29.P145-146：systemctl参数的用途。start, stop, restart, status,enable, disable,is-enabled参数的用途。（单选）

`systemctl` 命令参数的用途

`systemctl` 是用于控制 `systemd` 系统和服务管理器的工具。`systemd` 是大多数现代 Linux 发行版的初始化系统和服务管理器，管理系统的启动、服务、进程等。以下是常用的 `systemctl` 命令及其参数的说明：

1. start

- 启动一个服务。
- 用法示例：

```
systemctl start <service-name>
```

  - 启动

```
httpd
```

    服务：

```
systemctl start httpd
```

2. stop

- 停止一个正在运行的服务。
- 用法示例：

```
systemctl stop <service-name>
```

  - 停止

```
httpd
```

    服务：

```
systemctl stop httpd
```

3. restart

- 重启一个服务，即先停止服务再启动它。

- 用法示例：

```
systemctl restart <service-name>
```

  - 重启

```
httpd
```

    服务：

```
systemctl restart httpd
```

4. status

- 显示服务的当前状态，包括是否正在运行、是否启用了自动启动、日志等信息。
- 用法示例：

```
systemctl status <service-name>
```

  - 查看

```
httpd
```

    服务的状态：

```
systemctl status httpd
```

5. enable

- 设置服务为开机启动，即使系统重启，服务也会自动启动。
- 用法示例：

```
systemctl enable <service-name>
```

  - 设置

```
httpd
```

    服务为开机启动：

```
systemctl enable httpd
```

6. disable

- 禁用服务的开机自启动，使得服务不会在系统重启时自动启动。
- 用法示例：

```
systemctl disable <service-name>
```

- 禁用

  ```
  httpd
  ```

  服务的开机自启动：

  ```
  systemctl disable httpd
  ```

7. is-enabled

- 检查服务是否已经设置为开机自启动。返回 `enabled` 或 `disabled`。
- 用法示例：

```
systemctl is-enabled <service-name>
```

- 查看

  ```
  httpd
  ```

  服务是否启用了开机自启动：

  ```
  systemctl is-enabled httpd
  ```

总结：

- **start**: 启动服务。
- **stop**: 停止服务。
- **restart**: 重启服务。
- **status**: 查看服务状态。
- **enable**: 设置服务为开机启动。
- **disable**: 禁用服务的开机启动。
- **is-enabled**: 查看服务是否设置为开机启动。

这些命令用于管理系统服务，帮助管理员控制服务的生命周期和启动状态。

30.P149：crontab的时间字段语法。参考表6-4，表最后2行有错误，正确的顺序是分时天月星期。（单选）

`crontab` 的时间字段语法

`crontab` 用于配置周期性任务（定时任务），其时间字段的语法包括 **分钟**、**小时**、**日期**、**月份** 和 **星期**。以下是每个字段的详细说明和有效值范围。

时间字段顺序

正确的字段顺序为：

```
分　时　日　月　星期
```

- **分钟**（分）：0 到 59
- **小时**（时）：0 到 23
- **日期**（日）：1 到 31
- **月份**（月）：1 到 12
- **星期**（星期几）：0 到 7，其中 0 和 7 都表示星期天

特殊字符

- `*`（星号）：表示“所有值”，例如，`*` 在分钟字段表示每分钟都执行任务。
- `,`（逗号）：用于指定多个值，例如，`1,5,10` 在小时字段表示在1点、5点和10点执行。
- `-`（短横线）：用于指定范围，例如，`1-5` 表示从1到5。
- `/`（斜杠）：用于指定增量，例如，`*/5` 在分钟字段表示每5分钟执行一次。

示例

1. **每小时的第15分钟执行任务**：

   ```
   15 * * * * command
   ```

2. **每天凌晨1点执行任务**：

   ```
   0 1 * * * command
   ```

3. **每月的1号和15号凌晨12点执行任务**：

   ```
   0 0 1,15 * * command
   ```

4. **每周一的凌晨3点执行任务**：

   ```
   0 3 * * 1 command
   ```

5. **每5分钟执行一次任务**：

   ```
   */5 * * * * command
   ```

错误的顺序说明

如果表格中的时间字段顺序错误，并且表明为 **"分时天月星期"**，应该修正为 **"分 时 日 月 星期"**，因为这是正确的 `crontab` 时间格式。

31.P155-156：日志服务器主配置文件/etc/rsyslog.conf语法：.、.=、.!和@

在 `rsyslog` 配置文件 `/etc/rsyslog.conf` 中，日志规则使用了不同的符号和语法来定义日志消息的转发和存储方式。以下是您提到的符号及其含义：

1. `.`（点号）

点号 `.` 用于表示 **所有设施**。在 `rsyslog.conf` 中，设施表示了系统日志的不同类型，如 `mail`、`kern`、`auth` 等。如果使用 `*`（星号）作为设施的值，表示匹配所有设施。

例如：

```
*.crit   /var/log/critlog
```

这表示 **所有设施**（`*`）中，所有 `crit` 紧急程度及以上的日志都会被记录到 `/var/log/critlog`。

2. `.=`

`.=` 用于指定 **特定的紧急程度**，即仅将指定紧急程度的日志存储到文件中。与 `.*` 不同，`.=err` 会将紧急程度为 `err` 的日志（而不是 `err` 及以上）存储到文件。

例如：

```
mail.=err  /var/log/mailerrlog
```

这表示 **mail 设施** 中 **仅仅是 err 紧急程度的日志** 会被记录到 `/var/log/mailerrlog`。

3. `.!`

`.!` 用于指定 **低于某个紧急程度的日志**。它表示会忽略指定紧急程度及其以上的日志，只记录低于指定级别的日志。例如，`kern.!crit` 表示仅记录 `kern` 设施中低于 `crit` 紧急程度的日志。

例如：

```
kern.!crit  /var/log/kernlog
```

这表示 **kern 设施** 中 **低于 crit 紧急程度** 的日志会被记录到 `/var/log/kernlog`。

4. `@`

`@` 用于将日志消息发送到远程日志服务器。通常用于指定日志服务器的地址，以便将日志远程传输。您可以使用 `@` 表示 UDP 协议，或者 `@@` 表示 TCP 协议。

例如：

```
kern.*  @192.168.10.2
```

这表示 **kern 设施** 中所有紧急程度的日志会被发送到 **192.168.10.2** 的日志服务器。

示例解析

1. **mail.err /var/log/maillog**
   这是表示 **mail 设施** 的 **err 紧急程度及以上的日志** 会被记录到 `/var/log/maillog` 文件。

2. **mail.* /var/log/maillog**
   这是表示 **mail 设施** 的所有紧急程度的日志都会被记录到 `/var/log/maillog` 文件。
3. ***.crit   /var/log/critlog**
   这是表示 **所有设施** 中 **crit 紧急程度及以上的日志** 会被记录到 `/var/log/critlog` 文件。
4. **kern.!crit /var/log/kernlog**
   这是表示 **kern 设施** 中 **低于 crit 紧急程度** 的日志会被记录到 `/var/log/kernlog` 文件。
5. **mail.=err /var/log/mailerrlog**
   这是表示 **mail 设施** 中 **仅仅是 err 紧急程度的日志** 会被记录到 `/var/log/mailerrlog` 文件。
6. **kern.* @192.168.10.2**
   这是表示 **kern 设施** 中所有紧急程度的日志会被发送到 **192.168.10.2** 的日志服务器。

总结

- **.**: 表示所有设施。
- **.=err**: 表示仅记录指定紧急程度（如 `err`）的日志。
- **.!crit**: 表示记录低于指定紧急程度（如 `crit`）的日志。
- **@**: 表示将日志发送到远程日志服务器。

32.第七章

常用系统维护命令及其功能如下：

1. **top**
   显示实时的进程信息，包括 CPU 使用率、内存使用率、进程占用情况等。
   用法：`top`
2. **free**
   显示系统的内存使用情况，包括已用内存、空闲内存、缓存等。
   用法：`free`
3. **lsblk**
   显示系统中所有的块设备（如硬盘、分区等）信息，包括设备名称、大小、挂载点等。
   用法：`lsblk`
4. **df**
   显示文件系统的磁盘空间使用情况，包括已用空间、空闲空间、挂载点等。
   用法：`df`
5. **last**
   显示最近用户的登录记录，包括用户名、登录时间、退出时间等信息。
   用法：`last`

这些命令是系统管理员和用户常用的命令，用于监控和维护系统状态。

33.DNS服务器的类型：主DNS服务器、辅助DNS服务器、唯缓存DNS服务器之间的区别。（单选）

DNS服务器的类型及其区别如下：

1. **主DNS服务器（Primary DNS Server）**
   主DNS服务器负责存储域名的正式记录，并且是域名解析的权威来源。它保存着域名的所有资源记录（如A记录、MX记录等），并直接回答查询请求。如果主DNS服务器不可用，解析将会失败。

2. **辅助DNS服务器（Secondary DNS Server）**
   辅助DNS服务器是主DNS服务器的备份服务器，它从主DNS服务器获取域名记录的副本（通过区域传送）。辅助DNS服务器的作用是增强DNS的可靠性和负载均衡。如果主DNS服务器不可用，辅助DNS服务器可以继续提供解析服务，但它不能更新自己的记录，必须依赖主DNS服务器进行区域传送。
3. **唯缓存DNS服务器（Caching DNS Server）**
   唯缓存DNS服务器的作用是缓存经过查询的DNS记录。当客户端请求解析某个域名时，唯缓存DNS服务器首先检查缓存中是否已有相关记录。如果缓存中有有效记录，它会直接返回结果；如果没有，它会向其他DNS服务器发起请求来获取并缓存解析结果。唯缓存DNS服务器并不存储原始的DNS记录，也不承担权威解析。

区别：

- **主DNS服务器**：保存原始域名记录，是权威服务器。
- **辅助DNS服务器**：保存从主DNS服务器传送过来的副本，是备份服务器。
- **唯缓存DNS服务器**：仅缓存查询结果，提供更快的解析服务，但不保存原始记录

示例1：/etc/named.conf 配置文件解析

1. **zone "hezeu.edu.cn" IN {**
   - 表示正在配置 `hezeu.edu.cn` 这个域的DNS设置。
2. **type master;**
   - 该区域的DNS服务器类型为主DNS服务器（Master）。它是该域名的权威服务器，存储域名的主要记录。
3. **allow-transfer {192.168.0.2;};**
   - 允许辅助DNS服务器（192.168.0.2）从该主服务器进行区域传送，即获取域名的所有记录。
4. **zone "hezeu.edu.cn" IN {**
   - 重新定义另一个 `hezeu.edu.cn` 区域，但这次是配置辅助DNS服务器的设置。
5. **type slave;**
   - 该区域的DNS服务器类型为辅助DNS服务器（Slave）。它从主DNS服务器获取该域的副本，并且在主DNS服务器不可用时提供解析服务。
6. **masters {192.168.0.1;};**
   - 定义主DNS服务器的IP地址（192.168.0.1）。辅助DNS服务器将从这个主服务器获取区域数据。

示例2：example.com 域的DNS记录文件解析

1. **$TTL 86400**
   - 设置该区域记录的默认生存时间（TTL，Time To Live）为86400秒（即1天）。如果没有为记录单独指定TTL，则使用该默认值。
2. **@ IN SOA ns1.example.com. root.example.com. (...)**
   - `@` 代表该域（ `example.com` ）。
   - `IN SOA` 指示该行是域的起始授权（SOA）记录， `ns1.example.com.` 为主DNS服务器， `root.example.com.` 为管理员的邮箱（ `root@example.com` ）。
   - 该括号中的值是SOA记录的参数：
     - **0**：序列号，通常是一个递增的数字，每次记录变化时需要增加。
     - **2D**：刷新间隔，表示辅助DNS服务器每两天检查一次主服务器的记录。

- **1H**：重试间隔，表示如果辅助服务器无法获取主服务器的记录，它将在1小时后重试。
- **1W**：过期时间，表示如果辅助服务器在1周内无法访问主服务器，则该记录被认为过期。
- **3H**：最小TTL，表示查询结果的最短缓存时间。

3. **@ IN NS ns1.example.com.**
   - 该行表示 `example.com` 的主DNS服务器是 `ns1.example.com.` 。
4. **@ IN NS ns2.example.com.**
   - 该行表示 `example.com` 的辅助DNS服务器是 `ns2.example.com.` 。
5. **@ IN NS ns3.example.com.**
   - 该行表示 `example.com` 的另一个辅助DNS服务器是 `ns3.example.com.` 。
6. **video IN NS ns4.example.com.**
   - 该行表示 `video.example.com` 子域的DNS管理被委派给 `ns4.example.com.` 。
7. **ns1 IN A 192.168.10.11 和 ns1 IN A 192.168.10.12**
   - `ns1` 是 `example.com` 的主DNS服务器，并且有两个IP地址： `192.168.10.11` 和 `192.168.10.12` ，用来实现负载均衡。
8. **mail IN A 192.168.10.151 和其他A记录**
   - `mail` 是 `example.com` 的邮件服务器，记录多个IP地址（ `192.168.10.151` ， `192.168.10.152` ， `192.168.10.153` ）用于冗余和负载均衡。
9. **@ IN MX 10 mail**
   - 该行表示 `example.com` 的邮件交换服务器（MX记录）为 `mail.example.com.` ，优先级为10。
10. **w IN CNAME www**
    - `w.example.com.` 是 `www.example.com.` 的别名（CNAME记录）。这意味着 `w.example.com` 指向 `www.example.com` 。

解释

1. **域example.com的主域名服务器的域名**是： `ns1.example.com` 。
2. **主域名服务器的IP地址**是： `192.168.10.11` ， `192.168.10.12` 。
3. **管理员的邮箱地址**是： `root@example.com` （SOA记录中的 `root.example.com` ，其中 `.` 被替换为 `@` ）。
4. **辅助域名服务器的域名**是： `ns2.example.com` ， `ns3.example.com` 。
5. **辅助域名服务器的IP地址**是： `192.168.10.100` ， `192.168.10.101` ， `192.168.10.200` 。
6. **默认生存期（TTL）**是： `86400秒` （1天）。
7. **mail.example.com主机的IP地址**有： `3个` （ `192.168.10.151` ， `192.168.10.152` ， `192.168.10.153` ）。
8. **子域video.example.com委派给的服务器**是： `ns4.example.com` （IP地址： `192.168.10.240` ）。
9. **辅助服务器拷贝记录的间隔**是： `2天` （由SOA记录的 `refresh` 参数指定）。
10. **拷贝记录失败后的重试时间**是： `1小时` （由SOA记录的 `retry` 参数指定）。
11. **查询user123456@example.com的邮箱服务器域名**是： `mail.example.com` 。
12. **w.example.com的别名**是： `www.example.com` （由CNAME记录指定）。